GRÖBNER BASES OVER ALGEBRAIC NUMBER FIELDS

DEREJE KIFLE BOKU, WOLFRAM DECKER, CLAUS FIEKER,
AND ANDREAS STEENPASS

Abstract. Although Buchberger's algorithm, in theory, allows us to compute Gröbner bases over any field, in practice, however, the computational efficiency depends on the arithmetic of the ground field. Consider a field $K = \mathbb{Q}(\alpha)$, a simple extension of $\mathbb{Q}$, where α is an algebraic number, and let $f \in \mathbb{Q}[t]$ be the minimal polynomial of α. In this paper we present a new efficient method to compute Gröbner bases in polynomial rings over the algebraic number field K. Starting from the ideas of Noro [11], we proceed by joining f to the ideal to be considered, adding t as an extra variable. But instead of avoiding superfluous S-pair reductions by inverting algebraic numbers, we achieve the same goal by applying modular methods as in [2, 3, 10], that is, by inferring information in characteristic zero from information in characteristic $p > 0$. For suitable primes p, the minimal polynomial f is reducible over $\mathbb{F}_p$. This allows us to apply modular methods once again, on a second level, with respect to the factors of f. The algorithm thus resembles a divide and conquer strategy and is in particular easily parallelizable. At current state, the algorithm is probabilistic in the sense that, as for other modular Gröbner basis computations, an effective final verification test is only known for homogeneous ideals or for local monomial orderings. The presented timings show that for most examples, our algorithm, which has been implemented in Singular [7], outperforms other known methods by far.

1. Introduction

From the theoretical point of view, Gröbner bases computations can be done over any field by using Buchberger's algorithm (see, for example, [1, 6, 8]). In particular, they can be performed over an algebraic number field, but the computation is often inefficient if the arithmetic operations in this field are used directly. Consider a simple extension $K = \mathbb{Q}(\alpha)$ of $\mathbb{Q}$. Let $f \in \mathbb{Q}[t]$ be the minimal polynomial of α. The algebraic number field K can be represented as the residue class ring $\mathbb{Q}[t]/\langle f\rangle$, and a Gröbner basis computation over K can then be reduced to one over $\mathbb{Q}$ by joining f to the ideal to be considered. Unfortunately, this method is not satisfactory in view of efficiency. One of the reasons for this is that over the field of rational numbers, we often suffer from coefficient swell. Various methods to avoid this have been investigated; the trace algorithm [13] and modular algorithms [2, 10] are successful in this direction. But using these approaches, we still have to deal with the complicated arithmetic in algebraic number fields, in particular with the computation of inverses.

In this paper we present a new efficient method to compute Gröbner bases over an algebraic number field. Starting from a polynomial ring over $\mathbb{Q}$ as explained above, we apply the modular methods for computing Gröbner bases discussed in [2, 3, 10] to pass to positive characteristic p. Choosing a set $\mathcal{P}$ of suitable prime numbers, see Definition 5.2, the image f_p of f in $\mathbb{F}_p[t]$ is, for $p \in \mathcal{P}$, reducible and square-free. We can thus again apply modular methods, with respect to the factors $f_{1,p}, \dots, f_{r_p,p}$ of f_p, passing to the rings $\mathbb{F}_p[t]/\langle f_{i,p}\rangle$. As above, we avoid

Key words and phrases. Gröbner bases, algebraic number fields, factorization, Chinese remainder theorem, modular algorithms.

computing in quotient rings by joining $f_{i,p}$ to the ideal to be considered. Having computed the corresponding reduced Gröbner basis for each of these factors, we first recombine the results to a set of polynomials G_p over $\mathbb{F}_p[t]/\langle f_p \rangle$ using Chinese remaindering for polynomials. In a second lifting step, the sets G_p, $p \in \mathcal{P}$, are then used to reconstruct a set of polynomials G over $\mathbb{Q}$, via Chinese remaindering for integers and rational reconstruction. Finally, we test whether G is indeed the reduced Gröbner basis of the input ideal. If not, we enlarge $\mathcal{P}$ and repeat the process.

In Section 2, we introduce some notation which is used throughout this article. The structure of the new method is outlined in Section 3. Since this method relies on the Chinese remainder algorithm applied to different domains, we shortly recall the relevant theoretical background in Section 4. The core part of the proposed algorithm is discussed in Section 5. Here we explain how modular methods are applied on different levels and why our approach is considerably faster than other known methods. The application of modular methods follows a well-known scheme, see [3]. For reference, we recall the relevant parts of this scheme in Section 6. An illustrating example is given in Section 7. Finally, Section 8 contains remarks on the implementation of the new method in SINGULAR [7] and timings comparing it to other approaches. The benchmark problems which we used for the timings are listed in the appendix.

2. NOTATION

Let $K = \mathbb{Q}(\alpha)$ be an algebraic number field and let $f \in \mathbb{Q}[t]$ be the minimal polynomial of the algebraic number α. Then every element of K can be written as a linear combination of elements in $\{1, \alpha, \alpha^2, \ldots, \alpha^{d-1}\}$ where $d = \deg f$. Hence we may regard every element of K as a polynomial in α with coefficients in $\mathbb{Q}$. Let $X = \{x_1, \ldots, x_n\}$ be a set of variables, and let t be an extra variable. Consider the polynomial rings $S = \mathbb{Q}(\alpha)[X]$, $T = \mathbb{Q}[X, t]$, and $\mathbb{Q}[t]$. Fix a global monomial ordering $\succ_1$ on the monoid of monomials $\mathrm{Mon}(X)$ and consider the product ordering $\succ_K := (\succ_1, \succ)$ on $\mathrm{Mon}(X, t)$, where $\succ$ is the global ordering on $\mathrm{Mon}(t)$. Note that this implies $X^a \succ_K t^b$ for all $a \in \mathbb{N}^n \setminus \{(0, \ldots, 0)\}$ and $b \in \mathbb{N}$.

Let $\widetilde{H} = \{g_1(X, t), \ldots, g_s(X, t)\}$ be a subset of T, let $I \subseteq S$ be the ideal generated by $H := \{g_1(X, \alpha), \ldots, g_s(X, \alpha)\}$, and let $\widetilde{I} \subseteq T$ be the ideal generated by $\widetilde{H} \cup \{f\}$. Furthermore, let $\widetilde{G} \subseteq T$ be the reduced Gröbner basis (see [8, Definition 1.6.2]) of $\widetilde{I}$ w.r.t. $\succ_K$. Let φ be the canonical homomorphism from T to S which leaves the x_i fixed and maps t to α. We will show, in Theorem 5.1, that the non-zero elements of $\varphi(\widetilde{G}) \subseteq S$ form the reduced Gröbner basis of I w.r.t. $\succ_1$.

For a carefully chosen prime p (see Definition 5.2) which does not divide any denominator of the coefficients of f and $g_1(X, t), \ldots, g_s(X, t)$, we consider the map from $\mathbb{Q}$ to $\mathbb{F}_p$ which sends $\frac{a}{b}$ to $ab^{-1} \in \mathbb{F}_p$. Applying this map to the coefficients, we write $f_p := (f \bmod p) \in \mathbb{F}_p[t]$ and $\widetilde{I}_p := \langle g_1(X, t)_p, \ldots, g_s(X, t)_p, f_p \rangle \subseteq \mathbb{F}_p[X, t]$. Furthermore, for a polynomial $q \in S$ and a set $G \subseteq S$, we use the following notation:
$\mathrm{lm}(q)$: the *leading monomial* of q,
$\mathrm{Lm}(G)$: the *set of leading monomials* of the elements in G,
$\mathrm{lc}(q)$: the *leading coefficient* of q,
$\mathrm{lt}(q)$: the *leading term* or *head* of q,
$\mathrm{tail}(q) := q - \mathrm{lt}(q)$: the *tail* of q.

3. STRUCTURE OF THE NEW METHOD

Noro [11] has presented a modified version of Buchberger's algorithm which computes Gröbner bases over an algebraic number field using the arithmetic in $\mathbb{Q}[t]/\langle f \rangle$.

Instead of computing in the ring $(\mathbb{Q}[t]/\langle f\rangle)[X]$, one might as well add the minimal polynomial f to the ideal to be considered and work over $\mathbb{Q}[X,t]$, see Theorem 5.1. In this situation, the elements of a reduced Gröbner basis are, except f itself, all monic in $(\mathbb{Q}[t])[X]$, that is, they are of the form $X^a +$ (lower terms), see the proof of Theorem 5.1. Noro noticed that during the execution of Buchberger's algorithm, many (superfluous) intermediate basis elements of the form $t^bX^a +$ (lower terms) are computed before a monic element $X^a +$ (lower terms) is generated. Of course, each additional basis element produces new S-pairs which usually make the subsequent computation inefficient. Noro has resolved this problem by making each generated basis element monic in $(\mathbb{Q}[t])[X]$ before it is added to the basis. For this, the inverse of an algebraic number has to be computed which is in general computationally expensive. Instead, we use a different approach to reduce the number of basis elements which are computed before a monic element $X^a +$ (lower terms) is generated.

The new method computes the reduced Gröbner basis of the input ideal in three steps: In the first step, for a suitable prime p such that $f_p \in \mathbb{F}_p[t]$ is reducible and square-free, see Definition 5.2, we compute the reduced Gröbner basis $\widetilde{G}_p$ of $\widetilde{I}_p$ over $\mathbb{F}_p$ w.r.t. $\succ_K$, as follows: Let $f_p = \prod_{1\le i\le r_p} f_{i,p}$ be the irreducible factorization of f_p over $\mathbb{F}_p$, with $r_p > 1$. Set $\widetilde{I}_{i,p} := \langle \widetilde{H}_p \cup \{f_{i,p}\}\rangle \subseteq \mathbb{F}_p[X,t]$. For each $i \in \{1,\ldots,r_p\}$, we compute the reduced Gröbner basis $\widetilde{G}_{i,p}$ of $\widetilde{I}_{i,p}$. Using the Chinese remainder algorithm for polynomials (see Algorithm 1 below), we determine a set of polynomials $\widetilde{G}_p \equiv \left(\widetilde{G}_{i,p} \setminus \{f_{i,p}\}\right) \mod f_{i,p}$ which together with f_p is the reduced Gröbner basis of $\widetilde{I}_p$ with high probability (see Remark 5.6). Note that, at this step of the algorithm, computing modulo the different factors of the minimal polynomial $f_{i,p}$ (by adding them to the ideal $\langle \widetilde{H}_p\rangle$) is, from the theoretical point of view, just the same as computing modulo several prime numbers, see Section 4.

In the second step, following [2, 10], we use the Chinese remainder algorithm for integers together with rational reconstruction to lift these results to the reduced Gröbner basis $\widetilde{G}$ of $\widetilde{I}$. In the last step, we lift $\widetilde{G}$ to a Gröbner basis G of I over K by mapping t to α (see Theorem 5.1).

The idea of the algorithm is based on the concepts of modular methods and univariate polynomial factorization over finite fields. For the former we need the Chinese remainder theorem.

4. Factorization and the Chinese Remainder Algorithm

The well-known Chinese remainder theorem is essential for our algorithm.

Theorem 4.1 ([14, Corollary 5.3])**.** *Let R be a Euclidean domain and let $m_1,\ldots, m_r \in R$ be coprime elements so that $\gcd(m_i,m_j) = 1$ for $0 \le i < j \le r$. Let $m = m_1\cdots m_r$ be the product of these elements. Then $R/\langle m\rangle$ is isomorphic to the product ring $R/\langle m_1\rangle \times \ldots \times R/\langle m_r\rangle$ via the isomorphism*

$$R/\langle m\rangle \to R/\langle m_1\rangle \times \ldots \times R/\langle m_r\rangle\,,$$
$$a \mapsto (a \bmod m_1,\ldots,a \bmod m_r)\,.$$

For our purpose, we need this theorem in the following two incarnations.

Corollary 4.2. *Let $p_1,\ldots,p_k$ be distinct prime numbers, and let $N = p_1\cdots p_k$ be their product. Then we have the following isomorphism:*

$$\mathbb{Z}/\langle N\rangle \cong \mathbb{F}_{p_1} \times \ldots \times \mathbb{F}_{p_k}\,.$$

The second application of the Chinese remainder theorem refers to univariate polynomial rings over finite fields.

Corollary 4.3. *Let $f_{1,p}, \ldots, f_{r_p,p} \in \mathbb{F}_p[t]$ be pairwise coprime polynomials, and let $f_p = f_{1,p} \cdots f_{r_p,p}$ be their product. Then we have the ring isomorphism*

$$\mathbb{F}_p[t]/\langle f_p \rangle \cong \mathbb{F}_p[t]/\langle f_{1,p} \rangle \times \ldots \times \mathbb{F}_p[t]/\langle f_{r_p,p} \rangle \,.$$

The proof of Theorem 4.1 is constructive (see [14, Theorem 5.2, Corollary 5.3]) and yields the Chinese remainder algorithm. For reference, we state it here in the form of Corollary 4.3, see Algorithm 1.

Algorithm 1 Chinese Remainder Algorithm (CRA) for polynomials

```
Input: $q_1, \ldots, q_{r_p} \in \mathbb{F}_p[t]$, $f_{1,p}, \ldots, f_{r_p,p} \in \mathbb{F}_p[t]$ pairwise coprime.
Output: $g \in \mathbb{F}_p[t]$ such that $g \equiv q_i \mod f_{i,p}$ for $1 \leq i \leq r_p$.
$g \longleftarrow 0$
$f_p \longleftarrow \prod_{1 \leq i \leq r_p} f_{i,p}$
for $i = 1, \ldots, r_p$ do
    $h_i \longleftarrow \dfrac{f_p}{f_{i,p}}$
    by the Extended Euclidean Algorithm [14, Algorithm 3.14], compute $s_i, t_i \in \mathbb{F}_p[t]$ such that
        $$s_i h_i + t_i f_{i,p} = 1$$
    $c_i \longleftarrow \mathrm{NF}(q_i s_i, f_{i,p})$
    ($c_i$ is the remainder in $\mathbb{F}_p[t]$ on dividing $q_i s_i$ by $f_{i,p}$)
    $g \longleftarrow g + c_i h_i$
return $g$
```

Remark 4.4.

a) *Since $c_i h_i \equiv 0 \mod f_{j,p}$ for $j \neq i$ and $c_i h_i \equiv q_i s_i h_i \equiv q_i \mod f_{i,p}$, we have*

$$g \equiv c_i h_i \equiv q_i \mod f_{i,p} \,.$$

Hence, the algorithm works correctly.

b) *Although stated here for $\mathbb{F}_p[t]$, Algorithm 1 works for polynomial rings over any ground field.*

c) *Instead of $q_1, \ldots, q_{r_p} \in \mathbb{F}_p[t]$, Algorithm 1 can also be applied coefficientwise to polynomials with coefficients in $\mathbb{F}_p[t]$.*

5. Gröbner Bases using Factorization and Modular Methods

As Noro does (see [11, Theorem 1]), we rely on the following result whose proof we give for the lack of reference.

Theorem 5.1. *Let $\widetilde{G}$ be the reduced Gröbner basis of $\widetilde{I}$ w.r.t. $\succ_K$. Then $(\widetilde{G} \setminus \{f\})|_{t=\alpha}$ is the reduced Gröbner basis of I w.r.t. $\succ_1$.*

Consider the ring homomorphism

$$\varphi : T \longrightarrow S,\; t \longmapsto \alpha,\; x_i \longmapsto x_i \,.$$

Since φ is the identity map on $\mathbb{Q}[X]$, we get an isomorphism

$$S \cong T/\langle f \rangle \,.$$

Clearly, $\varphi(\widetilde{I}) = I$. We are now ready to prove Theorem 5.1.

Proof. Without loss of generality, we may assume that $\widetilde{I} \neq \langle 1 \rangle$. Let

$$\widetilde{G} = \{m_1(X,t), \dots, m_a(X,t), m_{a+1}(X,t)\}$$

be the reduced Gröbner basis of $\widetilde{I}$. We first prove that $f \in \widetilde{G}$. Suppose $f \notin \widetilde{G}$. Then there exists a non-zero non-constant polynomial $f' \in \widetilde{G} \cap \mathbb{Q}[t]$ with $\deg f' < \deg f$. Hence

$$I = \varphi(\widetilde{I}) = \langle \varphi(f'), \varphi(\widetilde{G} \setminus \{f'\}) \rangle = \langle 1 \rangle$$

since $\varphi(f')$ is invertible in S. This implies $\widetilde{I} = \langle 1 \rangle$, a contradiction. So, $f = m_i(X,t)$ for some i, say $i = a+1$. Then we have

$$\begin{aligned}\varphi(\widetilde{G} \setminus \{f\}) &= \{m_1(X,\alpha), \dots, m_a(X,\alpha)\} \\ &= (\widetilde{G} \setminus \{f\})|_{t=\alpha} =: G\,.\end{aligned}$$

The result follows easily once we show that the leading coefficient of $m(X,t)$, considered as an element in the polynomial ring $\mathbb{Q}[t]$, is equal to 1 for all $m(X,t) \in \widetilde{G} \setminus \{f\}$. To prove this statement, suppose there is an index $1 \leq j \leq a$ such that $\operatorname{lt}(m_j(X,t)) = c \cdot X^\delta$ with $c \in \mathbb{Q}[t]$ and $\deg c > 0$. Clearly, c is monic. Write

$$m_j(X,t) = c \cdot X^\delta + V(X,t)$$

where $V(X,t) = \operatorname{tail}(m_j(X,t))$, which implies that $V(X,t)$ does not contain any term divisible by X^δ. We have $\deg c < \deg f$ and therefore $\gcd(c,f) = 1$ since f is irreducible. Thus, by the extended Euclidean algorithm (see [14, Algorithm 3.14]), there exist $a, b \in \mathbb{Q}[t]$ such that $a \cdot c + b \cdot f = 1$. Considering the polynomial $a \cdot m_j(X,t) + b \cdot f \cdot X^\delta$, we have

$$\begin{aligned}\langle \widetilde{G} \rangle \ni{}& a \cdot m_j(X,t) + b \cdot f \cdot X^\delta \\ &= (a \cdot c + b \cdot f) \cdot X^\delta + a \cdot V(X,t) \\ &= X^\delta + a \cdot V(X,t) =: F(X,t)\,.\end{aligned}$$

But $\operatorname{lt}(F(X,t)) = X^\delta$ divides $c \cdot X^\delta = \operatorname{lt}(m_j(X,t))$ which is a contradiction to the choice of $\widetilde{G}$. □

The notion of primes which are *admissible of type* A w.r.t. a monic irreducible polynomial, which is essential for our algorithm, is defined as follows:

Definition 5.2. Let $f \in \mathbb{Q}[t]$ be as given above. Let p be a prime not dividing any numerator or any denominator of the coefficients occurring in f. We say that p is *admissible of type* A w.r.t. f if f_p is reducible and square-free over $\mathbb{F}_p$. In this case, we write f_p as $f_p = \prod_{1 \leq i \leq r_p} f_{i,p}$.

For a non-zero polynomial $g \in T$ considered as a polynomial in X over $\mathbb{Q}[t]$, that is, $g \in (\mathbb{Q}[t])[X]$, let S_g be the set of all distinct coefficients (in $\mathbb{Q}[t]$) of g of degree greater than or equal to 1. That is,

$$S_g = \left\{ \operatorname{lc}_{\mathbb{Q}[t]}(u) \mid u \text{ is a term of } g \text{ with } \deg(\operatorname{lc}_{\mathbb{Q}[t]}(u)) \geq 1 \right\}\,.$$

With notation as above, the notion of primes which are *admissible of type* B w.r.t. a monic irreducible polynomial and a set of polynomials is defined as follows:

Definition 5.3 (Weak version)**.** Let $\widetilde{H} = \{g_1(X,t), \dots, g_s(X,t)\}$ be as given above. Let p be a prime not dividing any numerator or any denominator of the coefficients occurring in $\widetilde{H}$. We say that p is *admissible of type* B w.r.t. f and $\widetilde{H}$ if p is admissible of type A w.r.t. f and if, for each g in $\widetilde{H}$, none of the elements in S_g is divisible by any of the factors of f_p over $\mathbb{F}_p$.

To see the relevance of this definition, consider the ideal

$$J = \langle x^2 + xy + t, x + y + t - 1\rangle =: \langle h_1, h_2\rangle \subseteq \mathbb{Q}[x, y, t]$$

and the minimal polynomial $f = t^3 + t + 1$. If $p = 3$, then $f_p \equiv (t-1)(t^2+t-1) =: f_{1,p} \cdot f_{2,p} \mod p$ and, using the degree reverse lexicographic ordering with $x \succ y$, the reduced Gröbner bases of the ideals $J_p + \langle f_{1,p}\rangle$ and $J_p + \langle f_{2,p}\rangle$ in $\mathbb{F}_p[x, y, t]$ are $\{1\}$ and $\{t^2 + t - 1, y + 1, x + t + 1\}$, respectively. In this case, Algorithm 1 cannot be applied since the sizes of these sets do not fit. The calculation suggests that the reason for this is that the element $t - 1 \in S_{h_2}$ vanishes when reduced w.r.t. the set $\{t - 1, t^2 + t - 1\}$.

Next, consider the ideal $J' = \langle x^2 + xy + t, t^2x + y\rangle =: \langle g_1, g_2\rangle$. Here, the reduced Gröbner bases of the ideals $J'_p + \langle f_{1,p}\rangle$ and $J'_p + \langle f_{2,p}\rangle$ are $\{1\}$ and $\{t^2 + t - 1, x + yt - t, y^2 - 1\}$, respectively. Again the sizes of these sets do not coincide, hence, we still cannot apply Algorithm 1. Moreover, none of the coefficients in S_{g_1} and S_{g_2} is divisible by either $f_{1,p}$ or $f_{2,p}$ which shows that the condition in Definition 5.3 is not sufficient. Indeed, the element $t^2 \in S_{g_2}$ vanishes when reduced w.r.t. the set $\{t^2 + t - 1, t - 1\}$. Therefore, we may impose a stronger condition by saying that for all $g \in \widetilde{H}$ none of the elements in S_g vanishes when reduced w.r.t. the set $\{f_{1,p}, \dots, f_{r_p,p}\}$ (in some order) and thus reduce the probability that the reconstruction fails. In the following example we see that this condition is still not sufficient.

Consider the ideal $J'' = \langle x^2 + xy + t, tx + y + t\rangle =: \langle k_1, k_2\rangle$. The reduced Gröbner bases of the ideals $J''_p + \langle f_{1,p}\rangle$ and $J''_p + \langle f_{2,p}\rangle$ are $\{t - 1, x - 1, y - 1\}$ and $\{t^2 + t - 1, x + yt - y + t + 1, y^2 + yt + y + t - 1\}$, respectively. Although none of the elements in S_{k_1} and S_{k_2} vanishes when reduced w.r.t. the set $\{t^2 + t - 1, t - 1\}$, and the sizes of these sets coincide, we see that applying Algorithm 1 yields $\{t^2 - t + 1, x - 1, y^2t^2 + y^2t - y^2 + yt^2 + yt + t^2 + t + 1\}$ which is not the desired result because the reduced Gröbner basis of $J''_p + \langle f_p\rangle$ is $\{t^2 + t - 1, y + 1, x + t + 1\}$. In practice, however, it is very unlikely that this case happens. It is, nevertheless, important to address this problem. A possible way to handle this difficulty is to refine Definition 5.3 as follows:

Definition 5.4 (Strong version)**.** Let f and $\widetilde{H} = \{g_1(X, t), \dots, g_s(X, t)\}$ be as given above. Let p be an a prime which is admissible of type A w.r.t. f, and write $f = f_{1,p} \cdots f_{r_p,p}$ as in Definition 5.2. Suppose that p does not divide any numerator or any denominator of the coefficients occurring in $\widetilde{H}$. For $i = 1, \dots, r_p$, set $\widetilde{I}_{i,p} := \langle \widetilde{H}_p \cup \{f_{i,p}\}\rangle$, and let $\widetilde{G}_{i,p}$ be the reduced Gröbner basis of the ideal $\widetilde{I}_{i,p}$. We say that p is *admissible of type* B w.r.t. f and $\widetilde{H}$ if for all indices i, j with $i \neq j$

a) the sizes of $\widetilde{G}_{i,p}$ and $\widetilde{G}_{j,p}$ coincide, and
b) $\mathrm{Lm}(\widetilde{G}_{i,p} \setminus \{f_{i,p}\}) = \mathrm{Lm}(\widetilde{G}_{j,p} \setminus \{f_{j,p}\})$.

In the above examples, the prime number 3 is not admissible of type B w.r.t. $t^3 + t + 1$ and the generators of each of the ideals J, J' and J'' in the sense of Definition 5.4. This is because in the first two cases, both conditions of this definition are violated whereas in the third case, the second condition is not satisfied. For the rest of our discussion we use the strong version of this definition.

We now turn our attention to the notion of *lucky primes*:

Definition 5.5 ([10])**.** Let $\widetilde{I}$ be an ideal given as above and let p be a prime number. Furthermore, let $\widetilde{G}$ be the reduced Gröbner basis of $\widetilde{I}$ and let $\widetilde{G}_p$ be the reduced Gröbner basis of $\widetilde{I}_p$. Then p is called *lucky* for $\widetilde{I}$ if and only if $\mathrm{Lm}(\widetilde{G}_p) = \mathrm{Lm}(\widetilde{G})$. Otherwise p is called *unlucky* for $\widetilde{I}$.

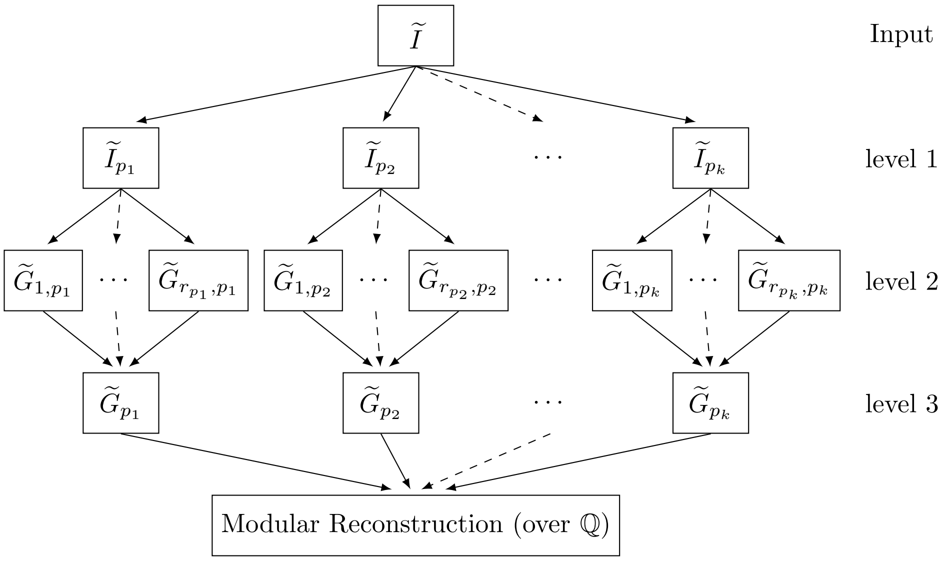

FIGURE 1. General scheme for the new algorithm

Since f is independent of X, we get, by Corollary 4.3, the isomorphism

$$\mathbb{F}_p[X,t]/\langle f_p\rangle \cong \mathbb{F}_p[X,t]/\langle f_{1,p}\rangle \times \ldots \times \mathbb{F}_p[X,t]/\langle f_{r_p,p}\rangle\,.$$

Remark 5.6. *Let $\widetilde{I}$, $\widetilde{H}$, and f be as above. Let p be a prime which is both admissible of type* B *w.r.t. f and $\widetilde{H}$ as well as lucky for $\widetilde{I}$. We work over $\mathbb{F}_p[X,t]$ equipped with the product ordering $\succ_K$. Suppose a set of polynomials $\widetilde{G}_p$ is the reduced Gröbner basis of the ideal $\widetilde{I}_p$. For $i = 1,\ldots,r_p$, set $S_i := (\widetilde{G}_p \setminus \{f_p\}) \bmod f_{i,p} \subseteq \mathbb{F}_p[X,t]/\langle f_{i,p}\rangle$. Then for each i, the set $S_i \cup \{f_{i,p}\}$ is the reduced Gröbner basis of the ideal $\widetilde{I}_{i,p}$ (as in Definition 5.4) with high probability. Conversely, let $\widetilde{G}_{i,p}$ be the reduced Gröbner basis of $\widetilde{I}_{i,p}$. Let $\widetilde{G}'_p$ be the set of polynomials that is obtained by applying Algorithm 1 coefficient-wise to the input*

$$\Big((\widetilde{G}_{1,p} \setminus \{f_{1,p}\},\ldots,\widetilde{G}_{r_p,p} \setminus \{f_{r_p,p}\}),\ (f_{1,p},\ldots,f_{r_p,p})\Big)\,.$$

Then the set $\widetilde{G}'_p \cup \{f_p\}$ is the reduced Gröbner basis of the ideal $\widetilde{I}_p$ with high probability. Hence, we have $\widetilde{G}'_p \cup \{f_p\} = \widetilde{G}_p$ with high probability.

The main innovation of our new algorithm, which is illustrated in Figure 1, is as follows: Instead of computing the reduced Gröbner bases at level 1, our algorithm computes them at level 2. For the primes satisfying the conditions in Definition 5.4 (and only for those), the Chinese remainder algorithm for polynomials then combines these results at level 3. The ideals $\langle\widetilde{G}_{p_i}\rangle$ at this level are expected to be the same as the ideals $\widetilde{I}_{p_i}$ at level 1 with high probability (see Remark 5.6). The remaining parts of the computation are carried out in the same way as in the modular algorithms described in [10].

Now we give a brief description of the new algorithm. In the beginning, randomly choose a set $\mathcal{P}$ of prime numbers which are admissible of type A w.r.t. f. At level 2, given a prime $p \in \mathcal{P}$, factorize $f \in \mathbb{Q}[t]$ over $\mathbb{F}_p$ and compute, for each i, the reduced Gröbner basis $\widetilde{G}_{i,p}$ of the ideal $\widetilde{I}_{i,p}$ corresponding to the i-th factor. If the prime p is admissible of type B w.r.t. f and $\widetilde{H}$, then lift these results via Chinese remaindering for polynomials (at level 3) to obtain the reduced Gröbner basis $\widetilde{G}_p$ of $\widetilde{I}_p$ with high probability. Repeat this process for every prime $p \in \mathcal{P}$ which is admissible of type B, in the same way as in the modular algorithms in [10].

The main reason why the method to compute Gröbner bases over algebraic number fields described above is faster than other known methods, see Section 8, is that factorizing the minimal polynomial f in positive characteristic allows us to compute in rings with minimal polynomials of degree much less than $\deg f$: Experiments have shown that the performance of Gröbner basis computations over simple algebraic extensions depends heavily on the degree of the minimal polynomial. Additionally, the computations are carried out over finite fields which avoids the problem known as coefficient swell, and we do not directly use the computationally expensive arithmetic in K. Finally, the new method is a priori easily parallelizable.

6. Modular Algorithms

To compute the reduced Gröbner basis of the ideal $\widetilde{I}$, the modular algorithm described in [10] first chooses a set of primes $\mathcal{P}$ and computes the reduced Gröbner bases $\widetilde{G}_p$ of $\widetilde{I}_p$ for each $p \in \mathcal{P}$. It then uses the Chinese remainder algorithm and rational reconstruction to obtain the reduced Gröbner basis $\widetilde{G}$ over $\mathbb{Q}$ with high probability. Finally, it verifies the correctness of the result obtained in this way. One of the problems after computing the set of reduced Gröbner bases $\mathcal{GP} := \{\widetilde{G}_p \mid p \in \mathcal{P}\}$ is that $\mathcal{P}$ may contain unlucky primes. To deal with such unlucky primes, the following method is used, see [3]:

DeleteUnluckyPrimesSB ([10]): *We define an equivalence relation on* $(\mathcal{GP}, \mathcal{P})$ *by*

$$(\widetilde{G}_p, p) \sim (\widetilde{G}_q, q) :\Longleftrightarrow \mathrm{Lm}(\widetilde{G}_p) = \mathrm{Lm}(\widetilde{G}_q)\,.$$

Then the equivalence class of largest cardinality[1] *is stored in* $(\mathcal{GP}, \mathcal{P})$*, the others are deleted.*

Now, all $\widetilde{G}_p$, $p \in \mathcal{P}$, have the same set of leading monomials. Hence, we can apply the Chinese remainder algorithm for integers and the rational reconstruction algorithm to the coefficients of the Gröbner bases in $\mathcal{GP}$ to obtain a reduced Gröbner basis $\widetilde{G}$ of $\widetilde{I}$ with high probability. Since we cannot check, however, whether $\mathcal{P}$ is sufficiently large, a final verification step is needed. Since this may be expensive, especially if $\widetilde{I} \neq \langle \widetilde{G} \rangle$, we first perform a test in positive characteristic:

pTestSB ([10]): *We randomly choose a prime* $p \notin \mathcal{P}$ *which is admissible of type* B *w.r.t.* f *and* $\widetilde{H}$*. We test if including this prime in the set* $\mathcal{P}$ *would improve the result. That is, explicitly test whether* $\widetilde{I}$ *reduces to zero w.r.t* $\widetilde{G}$ *mapped to* $\mathbb{F}_p[X, t]$*, and vice-versa, whether* $\widetilde{G}$ *mapped to* $\mathbb{F}_p[X, t]$ *reduces to zero w.r.t.* $\widetilde{G}_p$*.*

The advantage of this test is that it accelerates the algorithm enormously. Algorithm 2 is a modified version of Algorithm 1 in [10] (which is implemented in Singular [7] in the library `modstd.lib` [9]), in the sense that we do apply modular methods not only once, but twice, where the second application is with respect to the factors of the minimal polynomial f.

Now, taking Theorem 5.1 into account, we can compute a Gröbner basis of an ideal in $K[X] = \mathbb{Q}(\alpha)[X]$ as in Algorithm 3: We first map α to t and join the minimal polynomial $f \in \mathbb{Q}[t]$ to the ideal to be considered. Then, after applying Algorithm 2, we only need to map t back to α to get a Gröbner basis of the input ideal.

[1] Here, we have to use a weighted cardinality count if Algorithm 2 requires more than one round of the loop, see [3, Remark 5.7].

Algorithm 2 Modified modular Gröbner bases algorithm over $\mathbb{Q}$

```
Input: an ideal $\widetilde{I} = \langle \widetilde{H}, f \rangle \subseteq T = \mathbb{Q}[X,t]$ where $\widetilde{H} = \{g_1(X,t), \ldots, g_s(X,t)\}$ and
    $f \in \mathbb{Q}[t]$ is irreducible.
Output: $\widetilde{G} \subseteq T$, a Gröbner basis of $\widetilde{I}$ w.r.t. $\succ_K$.
 1: choose $\mathcal{P}$, a set of random primes which are admissible of type A w.r.t. $f$
 2: $\mathcal{GP} \longleftarrow \{\}$
 3: loop
 4:     for $p \in \mathcal{P}$ do
 5:         factorize $f_p \in \mathbb{F}_p[t]$ into irreducible factors $f_p = \prod_{1 \le i \le r_p} f_{i,p}$
 6:         for $i = 1, \ldots, r_p$ do
 7:             $\widetilde{I}_{i,p} \longleftarrow \langle \widetilde{H}_p \cup \{f_{i,p}\} \rangle \subseteq \mathbb{F}_p[X,t]$
 8:             compute the reduced Gröbner basis $\widetilde{G}_{i,p}$ of $\widetilde{I}_{i,p}$ w.r.t. $\succ_K$
 9:         if $p$ is admissible of type B w.r.t. $f$ and $\widetilde{H}$ over $\mathbb{F}_p$ then
10:             apply Algorithm 1 coefficient-wise to the input $\Big(\big(\widetilde{G}_{1,p} \setminus \{f_{1,p}\}, \ldots,$
                $\widetilde{G}_{r_p,p} \setminus \{f_{r_p,p}\}\big),\ \big(f_{1,p}, \ldots, f_{r_p,p}\big)\Big)$ to obtain a set of polynomials $\widetilde{G}_p \subseteq$
                $\mathbb{F}_p[X,t]$
11:             $\widetilde{G}_p \longleftarrow \widetilde{G}_p \cup \{f_p\}$
12:         else
13:             $\widetilde{G}_p \longleftarrow 0$
14:         $\mathcal{GP} \longleftarrow \mathcal{GP} \cup \{\widetilde{G}_p\}$
15:     $(\mathcal{GP}, \mathcal{P}) \longleftarrow$ DeleteUnluckyPrimesSB$(\mathcal{GP}, \mathcal{P})$
16:     lift $(\mathcal{GP}, \mathcal{P})$ to $\widetilde{G} \subseteq T$ by applying the Chinese remainder algorithm and the
        Farey rational map
17:     if pTestSB$(\widetilde{I}, \widetilde{G}, \mathcal{P})$ then
18:         if $\widetilde{I}$ reduces to zero w.r.t. $\widetilde{G}$ then
19:             if $\widetilde{G}$ is the reduced Gröbner basis of $\langle \widetilde{G} \rangle$ then
20:                 return $\widetilde{G}$
21:     enlarge $\mathcal{P}$
```

Algorithm 3 Modular Gröbner basis algorithm over $K = \mathbb{Q}(\alpha)$ (`nfmodStd`)

```
Input: $I = \langle g_1(X,\alpha), \ldots, g_s(X,\alpha) \rangle \subseteq S = K[X]$.
Output: $G \subseteq S$, a Gröbner basis of $I$ w.r.t. $\succ_1$.
map $I$ to $\langle \widetilde{H} \rangle$ via the map sending $\alpha$ to $t$
$\widetilde{I} \longleftarrow \langle \widetilde{H} \rangle + \langle f \rangle$
call Algorithm 2 to compute the reduced Gröbner basis $\widetilde{G}$ of $\widetilde{I}$ w.r.t. $\succ_K = (\succ_1,$
    $\succ)$
lift $\widetilde{G}$ to $G$ via the map sending $t$ to $\alpha$
return $G$
```

Algorithm 2 is probabilistic in the sense that the test in lines 17 to 18 does not guarantee that $\langle \widetilde{G} \rangle = \widetilde{I}$. If I is homogeneous, however, the result G of Algorithm 3 can be verified along the lines of [2, Theorem 7.1]. With this test included, Algorithm 3 is deterministic.

Remark 6.1. *Some parts of Algorithm 2 are inherently parallelizable. In the current implementation, see Section 8, we could easily take advantage of this thanks to* Singular*'s parallel framework. We have, first of all, parallelized the for-loop starting in line 4. This corresponds to the modular computations on level 1, see Figure 1. Besides this, we also make use of parallelization for the selection of primes* *in line 1, for the application of the Farey rational map in line 16, and for the final test in line 19. The for-loop starting in line 6, which corresponds to the modular computations on level 2, is inherently parallelizable as well, but experiments have shown that a parallel implementation of this step does not yield any further speedup for our test cases.*

7. Example

The following example illustrates how the new algorithm works:

Consider the ideal $I = \langle x^2 + ay, axy - x + a\rangle \subset \mathbb{Q}(a)[x,y]$ where a is a zero of the polynomial $f = t^2 + 1 \in \mathbb{Q}[t]$. A Singular computation shows that the reduced Gröbner basis of I with respect to the degree reverse lexicographical ordering (`dp` in Singular) with $x \succ y$ is

$$\{y^2 + ax + ay,\ xy + ax + 1,\ x^2 + ay\}\,.$$

In the following, we show how this basis is obtained using our method: At level 1, let us choose $k = 2$ with $p_1 = 5$ and $p_2 = 13$. At level 2, we have $f_{p_1} \equiv (t-2)(t+2) \mod p_1$ and $f_{p_2} \equiv (t-5)(t+5) \mod p_2$. Now, corresponding to each factor, we compute, using Singular, the reduced Gröbner bases of the following ideals:

$$\begin{aligned}\widetilde{I}_{1,p_1} &= \langle x^2 + ty, txy - x + t, t - 2\rangle\,,\\ \widetilde{I}_{2,p_1} &= \langle x^2 + ty, txy - x + t, t + 2\rangle\,,\\ \widetilde{I}_{1,p_2} &= \langle x^2 + ty, txy - x + t, t - 5\rangle\,,\\ \widetilde{I}_{2,p_2} &= \langle x^2 + ty, txy - x + t, t + 5\rangle\,.\end{aligned}$$

```
> ring r = 5, (x,y,t), (dp(2),dp(1));
> ideal I1p1 = x2+ty, txy-x+t, t-2;
> ideal I2p1 = x2+ty, txy-x+t, t+2;
> option(redSB);
> ideal S1 = std(I1p1);
> S1;
  S1[1]=t-2
  S1[2]=y2+2x+2y
  S1[3]=xy+2x+1
  S1[4]=x2+2y
> ideal S2 = std(I2p1);
> S2;
  S2[1]=t+2
  S2[2]=y2-2x-2y
  S2[3]=xy-2x+1
  S2[4]=x2-2y
```

The Chinese remainder algorithm for polynomials combines these results at level 3 to obtain the reduced Gröbner basis of $\widetilde{I}_{p_1}$ with high probability, as follows:

```
> list l = S1, S2;
> list m = t-2, t+2;
  // CRA for polynomials (coefficient-wise):
> ideal G1p1 = chinrempoly(l, m);
> Gp1;
  Gp1[1]=t2+1
  Gp1[2]=y2+xt+yt
  Gp1[3]=xy+xt+1
  Gp1[4]=x2+yt
```

Similarly, the reduced Gröbner basis of $\widetilde{I}_{p_2}$, with high probability, is

```
> Gp2;
  Gp2[1]=t2+1
  Gp2[2]=y2+xt+yt
  Gp2[3]=xy+xt+1
  Gp2[4]=x2+yt
```

It is not hard to see that the primes p_1 and p_2 are admissible of type B w.r.t. f and $\widetilde{H} = \{x^2 + ty, txy - x + t\}$. Furthermore, it is also clear that they are lucky primes for $\widetilde{I} = \langle \widetilde{H}, f\rangle$. At this point we have to change the current base ring in Singular to characteristic zero in order to apply the Chinese remainder algorithm for integers and to pull the modular coefficients back to the rational numbers.

```
/* Chinese remaindering for integers */
> ring s = 0, (x,y,t), (dp(2),dp(1));
> list l = imap(r, Gp1), imap(r, Gp2);
> intvec m = 5, 13;
> ideal j = chinrem(l, m);
> j;
  j[1]=t2+1
  j[2]=y2+xt+yt
  j[3]=xy+xt+1
  j[4]=x2+yt
/* rational reconstruction */
> j = farey(j, 5*13);
> j;
  j[1]=t2+1
  j[2]=y2+xt+yt
  j[3]=xy+xt+1
  j[4]=x2+yt
```

Note that the computed result already coincides with the reduced Gröbner basis stated above. To simplify the presentation, we therefore skip some of the steps in Algorithm 2, such as the final test. However, we have to map the result back to the ring $\mathbb{Q}(a)[x, y]$ in Singular:

```
> ring sr = (0,a), (x,y,t), (dp(2),dp(1));
> minpoly = a2+1;
> ideal G = imap(s, j);
> G = subst(G, t, a);
> G = simplify(G, 2); // erase the zero entries
> G; // G is the reduced Groebner basis of I
  G[1]=y2+ax+ay
  G[2]=xy+ax+1
  G[3]=x2+ay
```

Thus we get the same result as the one we mentioned at the beginning.

8. Implementation and Timings

We implemented Algorithm 3 in Singular in the library `nfmodstd.lib` [4] and compared its performance against the implementation of [10, Algorithm 1] in the Singular library `modstd.lib` (the command is `modStd`), the Singular command `std`, and the Magma [5, 12] command `GroebnerBasis`. For `modStd`, we added the minimal polynomial f to the given input ideal I (considered as an ideal in a polynomial ring over a polynomial ring) and computed the reduced Gröbner basis of the ideal $\widetilde{I} = \langle \widetilde{H}\rangle + \langle f\rangle$ w.r.t. $\succ_K$. For `GroebnerBasis` and `std`, we computed

Example			Magma	Singular				
Ideal	Min. Poly.	deg. of m_i	`GroebnerBasis`	`std`	`modStd`		`nfmodStd`	
					1 c.	32 c.	1 c.	32 c.
I1	m_1	2	1241.98	1.51	1.24	0.37	0.22	0.13
I2	m_2	5	error	70.55	19.59	4.79	1.89	0.61
I3a	m_3	7	-	0.90	143.79	9.34	3.27	0.51
I3b	m_3	7	-	314.00	11212.00	1118.78	97.43	19.23
I4	m_4	6	-	265.53	9163.38	567.03	686.01	99.41
I5	m_5	12	-	2061.95	3321.28	256.58	430.23	71.47
I6	m_6	2	2.93	8931.13	197.20	47.54	24.26	8.99
I7	m_7	8	-	0.90	2044.08	195.41	8.54	1.87
I8	m_8	7	-	15477.87	15274.97	4787.49	92.99	23.89

Table 1. Total running times in seconds for computing a Gröbner basis of the considered ideals with the corresponding minimal polynomial via `GroebnerBasis`, `std`, `modStd` and `nfmodStd`, using 1 core and 32 cores where applicable

the reduced Gröbner basis of the ideal I over an algebraic number field with the minimal polynomial f. Note that the implementation of our algorithm is internally linked with the existing implementation of Algorithm 1 in [10].

We have nine benchmark problems to demonstrate the superiority of our new algorithm (see appendix). The cyclic ideal C_n in n variables has become a benchmark problem for Gröbner basis techniques. For our algorithm, we have replaced the coefficients of this ideal by a random element in $\mathbb{Q}(a)$ where a is an algebraic number (see, for example, the ideal `I6` in the appendix). Some of the benchmark problems are chosen from [2, 11] (the ideals `I1` and `I2` are from [2], `I6` and `I7` are from [11]) where the coefficients are replaced by a random algebraic number. The minimal polynomials, selected for our computations, are:

$$\begin{aligned}
m_1 &= a^2 + 1\,,\\
m_2 &= a^5 + a^2 + 2\,,\\
m_3 &= a^7 - 7a + 3\,,\\
m_4 &= a^6 + a^5 + a^4 + a^3 + a^2 + a + 1\,,\\
m_5 &= a^{12} - 5a^{11} + 24a^{10} - 115a^9 + 551a^8 - 2640a^7\\
&\quad + 12649a^6 - 2640a^5 + 551a^4 - 115a^3 + 24a^2 - 5a + 1\,,\\
m_6 &= a^2 + 5a + 1\,,\\
m_7 &= a^8 - 16a^7 + 19a^6 - a^5 - 5a^4 + 13a^3 - 9a^2 + 13a\\
&\quad + 17\,,\ \text{and}\\
m_8 &= a^7 + 10a^5 + 5a^3 + 10a + 1\,.
\end{aligned}$$

With respect to these minimal polynomials, timings are conducted by using Singular 4.0.2 and Magma V2.21-2 on a Dell PowerEdge R720 machine with two Intel Xeon E5-2690 CPUs, 16 cores and 32 threads in total, 2.9-3.8 GHz, and 192 GB of RAM running the Gentoo Linux operating system.

The results are summarized in Table 1. Some of the computations in Magma did not finish within 12 hours. This is indicated by a dash (-). Note that in all those cases, the computation also occupied an excessive amount of memory, more than 100 GB at the point when we interrupted it. All timings are in seconds. We use the degree reverse lexicographical ordering (`dp` in SINGULAR) for all examples.

In our implementation, the number of primes which are chosen in line 1 of Algorithm 2 depends on the number of cores. For our timings, we started with 10 primes on one core and 25 primes on 32 cores. The runtime depends heavily on the splitting behaviour of the minimal polynomial modulo the chosen primes. Finding the optimal strategy for this is still under active research.

Remark 8.1. *We understand that Magma has no parallel version of the Gröbner basis algorithm which works over algebraic number fields. Therefore we have conducted the timings in Magma using one core only.*

From Table 1, we see that the SINGULAR commands `std` and `modStd` perform well in comparison to the Magma command `GroebnerBasis`. However, one can see that our algorithm `nfmodStd` is even much faster.

9. Acknowledgments

We would like to thank Gerhard Pfister for many fruitful discussions.

References

[1] W. W. Adams and P. Loustaunau. *An introduction to Gröbner bases.* American Mathematical Society, 1994.

[2] E. A. Arnold. Modular algorithms for computing Gröbner bases. *J. Symb. Comput.*, 35(4):403–419, 2003.

[3] J. Böhm, W. Decker, C. Fieker, and G. Pfister. The use of bad primes in rational reconstruction. To appear in *Math. Comp.*, 2015. `http://arxiv.org/abs/1207.1651`.

[4] D. K. Boku, W. Decker, and C. Fieker. `nfmodstd.lib`. A SINGULAR 4-0-2 library for computing Gröbner bases of ideals in polynomial rings over algebraic number fields, 2015. Included in SINGULAR 4-0-2 as `algemodstd.lib` and renamed to `nfmodstd.lib` for subsequent releases.

[5] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symb. Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[6] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra.* Springer, New York, third edition, 2007.

[7] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 4-0-2 – A computer algebra system for polynomial computations, 2015. `http://www.singular.uni-kl.de`.

[8] G.-M. Greuel and G. Pfister. *A Singular introduction to commutative algebra. With contributions by Olaf Bachmann, Christoph Lossen and Hans Schönemann.* Springer, Berlin, second extended edition, 2007.

[9] A. Hashemi, G. Pfister, H. Schönemann, A. Steenpass, and S. Steidel. `modstd.lib`. A SINGULAR 4-0-2 library for computing Gröbner bases of ideals using modular methods, 2014.

[10] N. Idrees, G. Pfister, and S. Steidel. Parallelization of modular algorithms. *J. Symb. Comput.*, 46(6):672–684, 2011.

[11] M. Noro. An efficient implementation for computing Gröbner bases over algebraic number fields. In *Mathematical software – ICMS 2006. Second international congress on mathematical software, Castro Urdiales, Spain, September 1–3, 2006. Proceedings*, pages 99–109. Springer, 2006.

[12] The Magma Group. The Magma Computational Algebra System V2.21-2, 2015. `http://magma.maths.usyd.edu.au`.

[13] C. Traverso. Gröbner trace algorithms. In P. Gianni, editor, *Symbolic and Algebraic Computation*, volume 358 of *Lecture Notes in Computer Science*, pages 125–138. Springer, 1989.

[14] J. von zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, Cambridge, third edition, 2013.

Appendix

The following are the benchmark problems used to demonstrate the efficiency of Algorithm 3. They are available in the source code of the SINGULAR library `nfmodstd.lib` [4].

(1)
```
ring R = (0,a), (x,y,z), dp;
minpoly = (a^2+1);
poly f1 = (a+8)*x^2*y^2+5*x*y^3+(-a+3)*x^3*z+x^2*y*z;
poly f2 = x^5+2*y^3*z^2+13*y^2*z^3+5*y*z^4;
poly f3 = 8*x^3+(a+12)*y^3+x*z^2+3;
poly f4 = (-a+7)*x^2*y^4+y^3*z^3+18*y^3*z^2;
ideal I1 = f1,f2,f3,f4;
```

(2)
```
ring R = (0,a), (x,y,z), dp;
minpoly = (a^5+a^2+2);
poly f1 = 2*x*y^4*z^2+(a-1)*x^2*y^3*z+(2*a)*x*y*z^2+7*y^3
          +(7*a+1);
poly f2 = 2*x^2*y^4*z+(a)*x^2*y*z^2-x*y^2*z^2+(2*a+3)*x^2*y*z
          -12*x+(12*a)*y;
poly f3 = (2*a)*y^5*z+x^2*y^2*z-x*y^3*z+(-a)*x*y^3+y^4
          +2*y^2*z;
poly f4 = (3*a)*x*y^4*z^3+(a+1)*x^2*y^2*z-x*y^3*z+4*y^3*z^2
          +(3*a)*x*y*z^3+4*z^2-x+(a)*y;
ideal I2 = f1,f2,f3,f4;
```

(3)
```
ring R = (0,a), (v,w,x,y,z), dp;
minpoly = (a^7-7*a+3);
poly f1 = (a)*v+(a-1)*w+x+(a+2)*y+z;
poly f2 = v*w+(a-1)*w*x+(a+2)*v*y+x*y+(a)*y*z;
poly f3 = (a)*v*w*x+(a+5)*w*x*y+(a)*v*w*z+(a+2)*v*y*z
          +(a)*x*y*z;
poly f4 = (a-11)*v*w*x*y+(a+5)*v*w*x*z+(a)*v*w*y*z+(a)*v*x*y*z
          +(a)*w*x*y*z;
poly f5 = (a+3)*v*w*x*y*z+(a+23);
ideal I3a = f1,f2,f3,f4,f5;
```

(4)
```
ring R = (0,a), (u,v,w,x,y,z), dp;
minpoly = (a^7-7*a+3);
poly f1 = (a)*u+(a+2)*v+w+x+y+z;
poly f2 = u*v+v*w+w*x+x*y+(a+3)*u*z+y*z;
poly f3 = u*v*w+v*w*x+(a+1)*w*x*y+u*v*z+u*y*z+x*y*z;
poly f4 = (a-1)*u*v*w*x+v*w*x*y+u*v*w*z+u*v*y*z+u*x*y*z
          +w*x*y*z;
poly f5 = u*v*w*x*y+(a+1)*u*v*w*x*z+u*v*w*y*z+u*v*x*y*z
          +u*w*x*y*z+v*w*x*y*z;
poly f6 = u*v*w*x*y*z+(-a+2);
ideal I3b = f1,f2,f3,f4,f5,f6;
```

(5)
```
ring R = (0,a), (w,x,y,z), dp;
minpoly = (a^6+a^5+a^4+a^3+a^2+a+1);
poly f1 = (a+5)*w^3*x^2*y+(a-3)*w^2*x^3*y+(a+7)*w*x^2*y^2;
poly f2 = (a)*w^5+(a+3)*w*x^2*y^2+(a^2+11)*x^2*y^2*z;
poly f3 = (a+7)*w^3+12*x^3+4*w*x*y+(a)*z^3;
poly f4 = 3*w^3+(a-4)*x^3+x*y^2;
ideal I4 = f1,f2,f3,f4;
```

(6)
```
ring R = (0,a), (w,x,y,z), dp;
minpoly = (a^12-5*a^11+24*a^10-115*a^9+551*a^8-2640*a^7
          +12649*a^6-2640*a^5+551*a^4-115*a^3+24*a^2-5*a+1);
poly f1 = (2*a+3)*w*x^4*y^2+(a+1)*w^2*x^3*y*z+2*w*x*y^2*z^3
          +(7*a-1)*x^3*z^4;
poly f2 = 2*w^2*x^4*y+w^2*x*y^2*z^2+(-a)*w*x^2*y^2*z^2
          +(a+11)*w^2*x*y*z^3-12*w*z^6+12*x*z^6;
```

```
    poly f3 = 2*x^5*y+w^2*x^2*y*z-w*x^3*y*z-w*x^3*z^2+(a)*x^4*z^2
              +2*x^2*y*z^3;
    poly f4 = 3*w*x^4*y^3+w^2*x^2*y*z^3-w*x^3*y*z^3
              +(a+4)*x^3*y^2*z^3+3*w*x*y^3*z^3+(4*a)*y^2*z^6-w*z^7
              +x*z^7;
    ideal I5 = f1,f2,f3,f4;
(7) ring R = (0,a), (u,v,w,x,y,z), dp;
    minpoly = (a^2+5*a+1);
    poly f1 = u+v+w+x+y+z+(a);
    poly f2 = u*v+v*w+w*x+x*y+y*z+(a)*u+(a)*z;
    poly f3 = u*v*w+v*w*x+w*x*y+x*y*z+(a)*u*v+(a)*u*z+(a)*y*z;
    poly f4 = u*v*w*x+v*w*x*y+w*x*y*z+(a)*u*v*w+(a)*u*v*z
              +(a)*u*y*z+(a)*x*y*z;
    poly f5 = u*v*w*x*y+v*w*x*y*z+(a)*u*v*w*x+(a)*u*v*w*z
              +(a)*u*v*y*z+(a)*u*x*y*z+(a)*w*x*y*z;
    poly f6 = u*v*w*x*y*z+(a)*u*v*w*x*y+(a)*u*v*w*x*z
              +(a)*u*v*w*y*z+(a)*u*v*x*y*z+(a)*u*w*x*y*z
              +(a)*v*w*x*y*z;
    poly f7 = (a)*u*v*w*x*y*z-1;
    ideal I6 = f1,f2,f3,f4,f5,f6,f7;
(8) ring R = (0,a), (w,x,y,z), dp;
    minpoly = (a^8-16*a^7+19*a^6-a^5-5*a^4+13*a^3-9*a^2+13*a+17);
    poly f1 = (-a^2-1)*x^2*y+2*w*x*z-2*w+(a^2+1)*y;
    poly f2 = (a^3-a-3)*w^3*y+4*w*x^2*y+4*w^2*x*z+2*x^3*z+(a)*w^2
              -10*x^2+4*w*y-10*x*z+(2*a^2+a);
    poly f3 = (a^2+a+11)*x*y*z+w*z^2-w-2*y;
    poly f4 = -w*y^3+4*x*y^2*z+4*w*y*z^2+2*x*z^3+(2*a^3+a^2)*w*y
              +4*y^2-10*x*z-10*z^2+(3*a^2+5);
    ideal I7 = f1,f2,f3,f4;
(9) ring R = (0,a), (t,u,v,w,x,y,z), dp;
    minpoly = (a^7+10*a^5+5*a^3+10*a+1);
    poly f1 = v*x+w*y-x*z-w-y;
    poly f2 = v*w-u*x+x*y-w*z+v+x+z;
    poly f3 = t*w-w^2+x^2-t;
    poly f4 = (-a)*v^2-u*y+y^2-v*z-z^2+u;
    poly f5 = t*v+v*w+(-a^2-a-5)*x*y-t*z+w*z+v+x+z+(a+1);
    poly f6 = t*u+u*w+(-a-11)*v*x-t*y+w*y-x*z-t-u+w+y;
    poly f7 = w^2*y^3-w*x*y^3+x^2*y^3+w^2*y^2*z-w*x*y^2*z
              +x^2*y^2*z+w^2*y*z^2-w*x*y*z^2+x^2*y*z^2+w^2*z^3
              -w*x*z^3+x^2*z^3;
    poly f8 = t^2*u^3+t^2*u^2*v+t^2*u*v^2+t^2*v^3-t*u^3*x
              -t*u^2*v*x-t*u*v^2*x-t*v^3*x+u^3*x^2+u^2*v*x^2
              +u*v^2*x^2+v^3*x^2;
    ideal I8 = f1,f2,f3,f4,f5,f6,f7,f8;
```

DEREJE KIFLE BOKU, DEPARTMENT OF MATHEMATICS, UNIVERSITY OF KAISERSLAUTERN, ERWIN-SCHRÖDINGER-STR., 67663 KAISERSLAUTERN, GERMANY
E-mail address: boku@mathematik.uni-kl.de

WOLFRAM DECKER, DEPARTMENT OF MATHEMATICS, UNIVERSITY OF KAISERSLAUTERN, ERWIN-SCHRÖDINGER-STR., 67663 KAISERSLAUTERN, GERMANY
E-mail address: decker@mathematik.uni-kl.de

CLAUS FIEKER, DEPARTMENT OF MATHEMATICS, UNIVERSITY OF KAISERSLAUTERN, ERWIN-SCHRÖDINGER-STR., 67663 KAISERSLAUTERN, GERMANY
E-mail address: fieker@mathematik.uni-kl.de

ANDREAS STEENPASS, DEPARTMENT OF MATHEMATICS, UNIVERSITY OF KAISERSLAUTERN, ERWIN-SCHRÖDINGER-STR., 67663 KAISERSLAUTERN, GERMANY
E-mail address: steenpass@mathematik.uni-kl.de